

CLAIMS

What is claimed is:

1. A method comprising:
building a queue having one or more drivers; and
executing the one or more drivers in the queue using a plurality of processors,
wherein the execution of drivers by each of the plurality of processors
includes:
determining whether there is a driver in the queue,
determining whether the driver is ready for execution, and
if the driver is ready for execution, executing the driver.
2. The method of claim 1, wherein a first processor of the plurality of processors is a
bootstrap processor.
3. The method of claim 2, wherein the plurality of processors includes one or more
application processors.
4. The method of claim 2, wherein if a second processor in the plurality of
processors determines that there are no drivers left in the queue, the second
processor goes to an idle state.
5. The method of claim 4, wherein if the first processor determines there are no
drivers left in the queue, the first processor:
waits until all other processors in the plurality of processors are in an idle state;
and

boots an operating system.

6. The method of claim 1, wherein the plurality of processors includes one or more logical processors.
7. The method of claim 1, wherein the execution of drivers in the queue further comprises removing a driver from the queue if the driver is ready for execution.
8. The method of claim 1, wherein the method is utilized in an extensible firmware interface (EFI).
9. The method of claim 1, wherein the plurality of drivers are executed in order.
10. The method of claim 9, wherein a first driver in the queue that has a dependency on a second driver in the queue is not executed until the second driver has been executed.
11. A processor comprising:
 - an execution unit; and
 - a first logical processor and a second logical processor, the first logical processor and the second logical processor utilizing the execution unit;
 - the first logical processor to build a queue having one or more drivers; and
 - the first logical processor and the second logical processor to execute the one or more drivers in the queue in parallel at least in part, wherein the execution of drivers includes:
 - determining whether there is a driver in the queue,
 - determining whether the driver is ready for execution, and

if the driver is ready for execution, executing the driver.

12. The processor of claim 11, wherein if the second logical processor determines there are no drivers left in the queue, the second logical processor is to enter an idle state.
13. The processor of claim 12, wherein if the first logical processor determines there are no drivers left in the queue, the first logical processor is to:
wait until the second processor is in an idle state; and
boot an operating system.
14. The processor of claim 11, wherein the processor operates concurrently with one or more other processors.
15. The processor of claim 11, wherein the execution of drivers in the queue further comprises removing the driver from the queue if the driver is ready for execution.
16. A system comprising:
a bootstrap processor;
one or more application processors;
a bus, the bootstrap processor and the one or more application processors being
coupled to the bus; and
a flash memory coupled to the bus;
wherein the bootstrap processor and the one or more application processors
execute a plurality of drivers in parallel at least in part, the execution of

the drivers by the bootstrap processor and each of the one or more application processors including:
determining whether there is a driver to be executed,
determining whether the driver is ready for execution, and
if the driver is ready for execution, executing the driver.

17. The system of claim 16, wherein if an application processor determines that there are no drivers left in the queue, the application processor is to enter an idle state.
18. The system of claim 17, if the first processor determines there are no drivers left in the queue, the first processor is to:
wait until all of the one or more application processors are in an idle state; and
boot an operating system.
19. The system of claim 16, the execution of drivers in the queue further comprises removing a driver from the queue if the driver is ready for execution.
20. The system of claim 19, the method is utilized in an extensible firmware interface (EFI).
21. A machine-readable medium having stored thereon data representing sequences of instructions that, when executed by a processor, cause the processor to perform operations comprising:
building a queue having one or more drivers; and

executing the one or more drivers in the queue using a plurality of processors,
wherein the execution of drivers by each of the plurality of processors
includes:
determining whether there is a driver in the queue,
determining whether the driver is ready for execution, and
if the driver is ready for execution, executing the driver.

22. The medium of claim 21, wherein a first processor of the plurality of processors is a bootstrap processor.
23. The medium of claim 22, wherein the plurality of processors includes one or more application processors.
24. The medium of claim 22, wherein if a second processor of the plurality of processors determines that there are no drivers left in the queue, the second processor goes to an idle state.
25. The medium of claim 24, wherein if the first processor determines there are no drivers left in the queue, the first processor is to:
wait until all other processors in the plurality of processors are in an idle state;
and
boot an operating system.
26. The medium of claim 21, wherein the plurality of processors includes one or more logical processors.

27. The medium of claim 21, wherein the execution of drivers in the queue further comprises removing a driver from the queue if the driver is ready for execution.
28. The medium of claim 21, wherein the method is utilized in an extensible firmware interface (EFI).
29. The medium of claim 21, wherein the plurality of drivers are executed in order.
30. The medium of claim 29, wherein a first driver in the queue that has a dependency on a second driver in the queue is not executed until the second driver has been executed.